

This Page Is Inserted by IFW Operations  
and is not a part of the Official Record

## **BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning documents *will not* correct images,  
please do not report the images to the  
Image Problem Mailbox.**



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE  
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re application of:

Lakshmi Ankireddipally, et al.

Serial No.: 09/574,335

Filed: May 19, 2000

Entitled: TRANSACTION DATA STRUCTURE FOR PROCESS COMMUNICATIONS  
AMONG NETWORK-DISTRIBUTED APPLICATIONS

MS Appeal Brief-Patents  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

Confirmation No. 8273

Group Art Unit No.: 2126

Examiner: Hoang, Phuong N.

**APPEAL BRIEF**

2126  
AFB/

**RECEIVED**

MAY 21 2004

Technology Center 2100

Sir:

This Appeal Brief is submitted in support of the Notice of Appeal filed on April 29, 2004.

**I. REAL PARTY IN INTEREST**

Sun Microsystems, Inc. is the real party in interest.

**II. RELATED APPEALS AND INTERFERENCES**

Appellant is unaware of any related appeals and interferences.

**III. STATUS OF CLAIMS**

Claims 21-30 are pending, have been finally rejected, and are the subject of this appeal.

15437-0511/P4620NP

Claims 21, 22, and 26-29 were rejected under 35 U.S.C. §102(e) as being anticipated by Berkowitz et al. (U.S. Patent No. 6,529,921).

Claim 23 was rejected under 35 U.S.C. §103(a) as being unpatentable over Berkowitz et al. in view of Chen (U.S. Patent No. 6,507,856).

Claims 24, 25, and 30 were rejected under 35 U.S.C. §103(a) as being unpatentable over Berkowitz et al. in view of Srinivasan (U.S. Patent No. 5,893,108).

#### **IV. STATUS OF AMENDMENTS**

No amendments have been made after the mailing of the Final Office Action.

#### **V. SUMMARY OF THE INVENTION**

In a network-distributed computing system, a plurality of nodes may be situated at various locations, and may be interconnected via a network to enable the nodes to communicate and to interact with each other. In such a system, each node may provide one or more services, and these services may be invoked via the network by one or more other nodes. Each service may be part of an overall transaction. That is, a transaction may require invoking services from a plurality of different nodes. Because of the distributed nature of this type of system, managing a transaction can be relatively complex.

In one embodiment of the present invention, there is provided a mechanism for facilitating the management of a transaction in a network-distributed system. Specifically, when a transaction is to be performed, a transaction instance data structure is produced. This data structure indicates a plurality of operations (e.g. services) constituting the transaction. The data structure also indicates a linking of the operations to indicate an operation performance order,

and further indicates conditioning logic data for changing the performance order of the operations. Basically, the data structure acts as a specification for the transaction.

After the data structure is produced, a number of steps are carried out for each of the operations indicated in the data structure. These steps include producing an operation request message indicating input data, sending the operation request message to a service application (executing, for example, on a node) to perform the operation using the input data, receiving an operation response message from the service application indicating output data from the operation, and determining a next operation to perform using the conditioning logic data in the data structure and the output data of the operation response message. By performing these steps for each operation in the transaction, the overall transaction is carried out.

This mechanism provides many different advantages. One such advantage is that it enables the entire transaction to be managed from a central point. Even if the operations of the transaction are performed by service applications situated at different nodes in a distributed-network system, it does not matter. This mechanism can still be used to manage the transaction. As a result, this embodiment of the present invention makes it possible to manage a transaction centrally and conveniently, even in a network-distributed system.

## **VI. ISSUES**

Issue #1: Whether the Examiner erred in rejecting claims 21, 22, and 26-29.

Issue #2: Whether the Examiner erred in rejecting claim 23.

Issue #3: Whether the Examiner erred in rejecting claims 24, 25, and 30.

## VII. GROUPING OF CLAIMS

Claims 21, 22, and 26-29 stand or fall together.

Claim 23 stands on its own.

Claims 24, 25, and 30 stand or fall together.

## VIII. ARGUMENTS

### A. The Examiner has erred in rejecting claims 21, 22, and 26-29

In the Final Office Action, the Examiner rejected claims 21, 22, and 26-29 under 35 U.S.C. §102(e) as being anticipated by Berkowitz et al. (U.S. Patent No. 6,529,921). In order for a reference to anticipate a claim, the reference must disclose each and every limitation of the claim. Berkowitz fails to do so with regard to claims 21, 22, and 26-29.

Independent claim 21 recites:

A computer-implemented method for performing a transaction comprising the steps of:  
producing a transaction instance data structure indicating a plurality of operations  
constituting a transaction; the transaction instance data structure indicating a  
linking of the plurality of operations to indicate an operation performance order;  
the transaction instance data structure further indicating conditioning logic data  
for changing the operation performance order such that the plurality of operations  
is capable of being performed in more than one possible order; and  
for each of the plurality of operations,  
producing an operation request message indicating input data for performing an  
operation;  
sending the operation request message to a service application to perform the operation  
using the input data;  
receiving an operation response message from the service application indicating output  
data from the operation; and  
determining a next operation to perform using the conditioning logic data and the output  
data of the operation response message.

Claim 21 provides an advantageous method for managing the performance of a transaction. The method may be applied in many contexts, including one in which a transaction

is carried out in a network-distributed system. Such a method is neither disclosed nor suggested by Berkowitz et al.

Instead, Berkowitz et al. discloses a method and apparatus for maintaining a coherent set of database tables (i.e. a coherent cache) among a plurality of nodes. In Berkowitz et al., when a new node joins the plurality of nodes (and hence, joins the "coherent cache"), one of the existing nodes is selected to be a source node. The task of the source node is to provide the new node with a set of database tables that are current and synchronized with the tables on all of the other nodes. To do so, the source node first provides the new node with the data that is currently in its database tables. While this data is being provided, the database tables may be updated by ongoing transactions. Thus, the source node also provides to the new node a "snapshot", which captures all of the active transactions (transactions that have not been committed or aborted) that involve the tables being synchronized. The new node can thereafter use the snapshot information to update the information in its tables, as needed. The new node is thus brought online into the coherent cache.

In rejecting claim 21, the Examiner interpreted (by citing Col. 14, lines 30-65 of Berkowitz et al.) the snapshot of Berkowitz et al. to be the "transaction instance data structure" recited in claim 21, and the changes included in the snapshot to be the "operations" recited in claim 21. In light of this interpretation, a number of points should be noted.

First, it should be noted that the changes in the snapshot do not truly constitute "operations" in that they do not call for any action to be performed. Rather, they are just sets of data that are provided to the new node. The new node may at some point decide to use the sets of data to update its tables, but the sets of data in and of themselves do not call for any action to

be performed. Thus, the changes included in the snapshot cannot be fairly read to be the "operations" recited in claim 21.

Another point to note is that, unlike the transaction instance data structure of claim 21, the snapshot of Berkowitz et al. does not indicate any linking between the changes to indicate a performance order. More particularly, there is no relationship specified between the changes to indicate any type of order between the changes. As far as the source node is concerned, the changes may be sent to the new node in any order. In support of the rejection, the Examiner cites Col. 6, lines 15-40 of Berkowitz et al., which discusses the arbitration process implemented by the nodes to determine an order in which to apply changes. The portion cited by the Examiner, however, has little to do with the snapshot process. In fact, as stated in Col. 8, lines 20-21, arbitration is frozen during the creation of the snapshot. Thus, none of the discussion provided in Col. 6, lines 15-40 pertaining to arbitration relates to the snapshot process. Overall, there is nothing in the snapshot process that indicates a linking of the changes to indicate a performance order.

Yet another point to note is that, unlike the transaction instance data structure of claim 21, the snapshot of Berkowitz et al. has no conditioning logic data that can be used to alter the performance order of the changes. As noted above, the snapshot has no linking information to indicate a performance order. Thus, it follows that the snapshot also has no conditioning logic data that can be used to change the performance order. In support of the rejection, the Examiner contends that the status of a transaction (commit or abort) can be read to be the conditioning logic data. Appellants disagree. Whether a transaction has been committed or aborted (i.e. whether the transaction is still active) determines whether a change is included in the snapshot. It does not determine, nor can it be used to change the order in which the changes are sent to the

new node. Berkowitz et al. does not disclose or suggest anything in the snapshot that can be fairly read to be conditioning logic data.

Yet another point to note is that, unlike claim 21, when the source node sends a change to the new node, the source node is not sending an "operation request" to the new node. Put another way, the source node is not asking the new node to perform any operation. Rather, it is simply sending the new node a set of data. Further, unlike claim 21, the source node does not receive anything from the new node. It certainly does not receive anything that even remotely resembles output data from an operation, as recited in claim 21. Since the source node is not asking the new node to perform any operation, it should come as no surprise that the source node does not receive any output data from the new node. Further, unlike claim 21, the source node does not determine a next operation to perform using the conditioning logic data and the output data. As noted above, the snapshot has no conditioning logic data, and the source node receives no output data from the new node. Thus, the source node cannot possibly determine a next operation using conditioning logic data and output data. In support of the rejection, the Examiner cites various portions of Berkowitz et al. that do not relate to the snapshot process. These portions cannot be read fairly into the snapshot process to reject claim 21.

Overall, the Examiner has failed to put together a logical and coherent showing that all of the limitations of claim 21 are disclosed by Berkowitz et al. Instead, the Examiner has plucked various pieces from different and unrelated aspects of Berkowitz et al., and has put them together in a tortured and arbitrary manner to try to arrive at something that resembles claim 21. This can be seen clearly in the Examiner's rejection of claim 21 (paragraph 5 of the Final Office Action).



Specifically, in support of the contention that Berkowitz et al. discloses a "transaction instance data structure", the Examiner cites Col. 14, lines 30-65, which discusses a snapshot. However, to show that this data structure indicates an operation performance order, the Examiner cites Col. 6, lines 15-40, a completely different part of the patent, which discusses an arbitration process unrelated to the snapshot. Thus, to support his contention that Berkowitz et al. discloses a transaction instance data structure that indicates an operation performance order, the Examiner pulls excerpts from very different portions of Berkowitz et al., which are eight columns apart, and which discuss very different topics having have very little to do with each other!

Proceeding further with the rejection, in support of his contention that Berkowitz et al. discloses receiving an operation response message indicating output data, the Examiner cites Col. 6, lines 35-45, which discusses the use of proposal and closure messages in an arbitration process. However, to show that this output data is used (along with the conditioning logic data) to determine a next operation to perform, the Examiner cites Col. 15, lines 18-59, which is nine columns removed, and which describes the manner in which transactions are applied in a synchronization process. The arbitration process and the synchronization process have very little to do with each other. Worse yet, the messages discussed in the first excerpt do not include any output data, and they are not even taken into account in the "next transaction" determination made in the second excerpt. Thus, there is no logical or technical link between the excerpts. Consequently, it makes no logical sense to link the teachings of one excerpt with that of the other. Nonetheless, that is exactly what the Examiner has done.

As the above discussion shows, rather than setting forth a logical and coherent showing that all of the limitations of claim 21 are disclosed by Berkowitz et al., the Examiner has cut and pasted together various different and unrelated teachings of Berkowitz et al. in an attempt to

arrive at the method of claim 21. This is clearly improper. Consequently, Appellants request that the Examiner's rejection of claim 21 be reversed.

Claims 22, 26, and 29 depend from claim 21. Claim 28 is an article of manufacture claim which is analogous to the method of claim 21. Claim 29 depends from claim 28. Appellants request that the rejection of these claims be reversed as well.

B. The Examiner erred in rejecting claim 23

In the Final Office Action, the Examiner rejected claim 23 under 103(a) as being unpatentable over Berkowitz et al. as applied to claim 21, in view of Chen (U.S. Patent No. 6,507,856). Claim 23 depends from claim 21, and hence, incorporates all of the limitations of claim 21. Appellants submit that the Examiner erred in making this rejection.

As argued above, Berkowitz et al., taken alone, fails to disclose many aspects of claim 21. These aspects are also not disclosed in Chen. Instead, the Examiner relies on Chen to show just the XML aspects of claim 23, namely, that the operation request message and the operation response message include XML tags. Thus, even combined, assuming for the sake of argument that it would have been obvious to combine the references, Berkowitz et al. and Chen still would not disclose all of the limitations of claim 23. For at least this reason, Appellants submit that claim 23 was rejected in error, and request that the rejection be reversed.

C. The Examiner erred in rejecting claims 24, 25, and 30

In the Final Office Action, the Examiner rejected claims 24, 25, and 30 under 103(a) as being unpatentable over Berkowitz et al. as applied to claim 21, in view of Srinivasan (U.S. Patent No. 5,893,108). Claims 23 and 24 depend from claim 21, and hence, incorporate all of

the limitations of claim 21. Claim 30 depends from claim 28, and hence, incorporates all of the limitations of claim 30. Appellants submit that the Examiner erred in making this rejection.

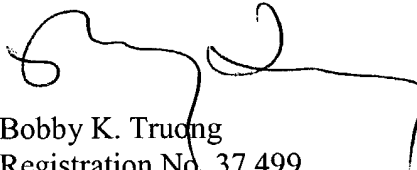
As argued above, Berkowitz et al., taken alone, fails to disclose many aspects of claims 21 and 28. These aspects are also not disclosed in Srinivasan. Instead, the Examiner relies on Srinivasan to show just that the transaction instance data structure may be a directed acyclic graph (DAG). Thus, even combined, assuming for the sake of argument that it would have been obvious to combine the references, Berkowitz et al. and Srinivasan still would not disclose all of the limitations of claims 24, 25, and 30. For at least this reason, Appellants submit that claims 24, 25, and 30 were rejected in error, and request that the rejection be reversed.

#### IX. CONCLUSION AND PRAYER FOR RELIEF

For the foregoing reasons, it is respectfully submitted that the Examiner erred in rejecting claims 21-30 under 35 U.S.C. §102(e) and §103(a). Consequently, Appellants respectfully request that the Honorable Board reverse the Examiner's rejections.

Respectfully submitted,

HICKMAN PALERMO TRUONG & BECKER LLP

  
Bobby K. Truong  
Registration No. 37,499

Date: May 13, 2004

1600 Willow Street  
San Jose, California 95125-5106  
Tel: (408) 414-1224  
Fax: (408) 414-1076

#### CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: MS Appeal Brief-Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450

on May 13, 2004

by   
Annette Jacobs

## CLAIMS APPENDIX

1    21. A computer-implemented method for performing a transaction comprising the  
2    steps of:  
3    producing a transaction instance data structure indicating a plurality of operations  
4       constituting a transaction; the transaction instance data structure indicating a  
5       linking of the plurality of operations to indicate an operation performance order;  
6       the transaction instance data structure further indicating conditioning logic data  
7       for changing the operation performance order such that the plurality of  
8       operations is capable of being performed in more than one possible order; and  
9    for each of the plurality of operations,  
10       producing an operation request message indicating input data for performing an  
11       operation;  
12       sending the operation request message to a service application to perform the  
13       operation using the input data;  
14       receiving an operation response message from the service application indicating  
15       output data from the operation; and  
16       determining a next operation to perform using the conditioning logic data and the  
17       output data of the operation response message.

1       22. The computer-implemented method of claim 21 for performing a  
2       transaction wherein the conditioning logic data indicates at least one of a  
3       mathematical expression, a function, and a variable data item; and wherein the

4 step of determining the next operation to perform using the conditioning logic  
5 data and the output data of the operation response message includes using the  
6 output data to evaluate the at least one of the mathematical expression, the  
7 function, and the variable data item.

1 23. The computer-implemented method of claim 21 for performing a  
2 transaction wherein the operation request message and the operation response  
3 message include extensible markup language (XML) tags indicating data  
4 items.

1 24. The computer-implemented method of claim 21 for performing a  
2 transaction wherein the transaction instance data structure is a directed acyclic  
3 graph (DAG) including a plurality of nodes; each operation being represented  
4 by a node; the nodes being arranged in the transaction instance DAG such that  
5 paths through the transaction instance DAG indicate the more than one  
6 possible order in which the plurality of operations may be performed; and  
7 wherein performing the transaction further includes traversing a path through  
8 the plurality of nodes of the transaction instance DAG.

1 25. The computer-implemented method of claim 24 for performing a  
2 transaction wherein the path through the graph is determined at runtime.

1        26.    The computer-implemented method of claim 21 for performing a  
2        transaction further including receiving a transaction request message indicating  
3        a request to perform the transaction from a requesting application residing on a  
4        first computer included in a distributed network; and wherein the service  
5        application resides on a second computer included in the distributed network.

1        27.    The computer-implemented method of claim 26 wherein the distributed  
2        network is the Internet.

1        28.    An article of manufacture comprising a data storage medium having  
2        computer readable instruction data embodied therein; the computer readable  
3        instruction data indicating instructions executed by a processor in a processor-  
4        controlled machine for managing transaction processing message flow among  
5        a plurality of requesting application programs and service application  
6        programs resident on a plurality of processor-controlled machines in a  
7        distributed network; the computer readable instructions in the article of  
8        manufacture comprising:  
9        a first portion of instructions which when executed causes the processor to produce  
10       a transaction instance data structure indicating a plurality of operations  
11       constituting a transaction; the transaction instance data structure indicating a  
12       linking of the plurality of operations to indicate an order of execution; the  
13       transaction instance data structure further indicating conditioning logic data

14 conditioning execution of at least one operation such that the plurality of  
15 operations is capable of being performed in more than one possible order; and  
16 a second portion of instructions which when executed causes the processor, for  
17 each of the plurality of operations, to produce an operation request message  
18 indicating input data for performing an operation, to send the operation request  
19 message to a service application to perform the operation using the input data,  
20 to receive an operation response message from the service application  
21 indicating output data from the operation, and to determine a next operation to  
22 perform using the conditioning logic data and the output data of the operation  
23 response message.

1 29. The article of claim 28 wherein the conditioning logic data indicates at  
2 least one of a mathematical expression, a function, and a variable data item;  
3 and wherein the second portion of instructions further includes a third portion  
4 of instructions which, when executed, causes the processor, for each of the  
5 plurality of operations, to use the output data to evaluate the at least one of the  
6 mathematical expression, the function, and the variable data item in order to  
7 determine the next operation to perform.

1 30. The article of claim 28 wherein the transaction instance data structure is a  
2 directed acyclic graph (DAG) including a plurality of nodes; each operation being  
3 represented by a node; the nodes being arranged in the transaction instance DAG

4       such that paths through the transaction instance DAG indicate the more than one  
5       possible order in which the plurality of operations may be performed; and wherein  
6       the article further includes a third portion of instructions which, when executed,  
7       causes the processor to traverse a path through the plurality of nodes of the  
8       transaction instance DAG.